Application Software Development Lab

College Of Engineering Trivandrum

Department of Computer Science

Dhanesh P S
S5 CSE Roll No:17

TVE18CS017

November 24, 2020

Contents

Chapter 1

Cycle 2 Lab Exam

1.1 Aim

Create a PL/SQL program to calculate the bill amount of a customer.

1.2 Table Creation

1.2.1 Command

```
CREATE TABLE customer (
    cust_id INT PRIMARY KEY,
    meter_type INT,
    previous_reading INT,
    current_reading INT,
    customer_type CHAR(2),
    bill_amount DECIMAL(10, 2),
    last_bill_payment CHAR(2)
);

INSERT INTO customer(cust_id, meter_type, previous_reading, current_reading, customer
VALUES
    (10005, 1, 3500, 3550, 'R', 'N'),
    (10004, 3, 4800, 4989, 'C', 'Y'),
    (10003, 1, 5600, 5700, 'I', 'Y'),
    (10008, 3, 5700, 6000, 'A', 'N'),
```

```
    (10002, 3, 4890, 5042, 'C', 'Y')
;

CREATE TABLE bill (
    cust_id INT,
    bill_date DATE,
    fixed_charge DECIMAL(10, 2),
    energy_charge DECIMAL(10, 2),
    sur_charge DECIMAL(10, 2),
    net_amount DECIMAL(10, 2)
);

INSERT INTO bill
VALUES
    (10005, '24/09/2020', 40, 82, 4.1, 126.1),
    (10005, '24/07/2020', 40, 70, 3.5, 113.5),
    (10008, '24/09/2020', 120, 187.5, 18.75, 326.25)
;
```

1.2.2 Output

```
postgres@dhanesh:~$ psql
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# CREATE TABLE customer (
postgres(#     cust_id INT PRIMARY KEY,
postgres(#     meter_type INT,
postgres(#     previous_reading INT,
postgres(#     current_reading INT,
postgres(#     customer_type CHAR(2),
postgres(#     bill_amount DECIMAL(10, 2),
postgres(#     last_bill_payment CHAR(2)
postgres(# );
CREATE TABLE
postgres=#
postgres=# INSERT INTO customer(cust_id, meter_type, previous_reading, curre
nt)
postgres-# VALUES
postgres-#     (10005, 1, 3500, 3550, 'R', 'N'),
postgres-#     (10004, 3, 4800, 4989, 'C', 'Y'),
postgres-#     (10003, 1, 5600, 5700, 'I', 'Y'),
postgres-#     (10008, 3, 5700, 6000, 'A', 'N'),
postgres-#     (10002, 3, 4890, 5042, 'C', 'Y')
postgres-# ;
INSERT 0 5
```

```
postgres=# CREATE TABLE bill (
postgres(#     cust_id INT,
postgres(#     bill_date DATE,
postgres(#     fixed_charge DECIMAL(10, 2),
postgres(#     energy_charge DECIMAL(10, 2),
postgres(#     sur_charge DECIMAL(10, 2),
postgres(#     net_amount DECIMAL(10, 2)
postgres(# );
CREATE TABLE
postgres=#
postgres=# INSERT INTO bill
postgres-# VALUES
postgres-#     (10005, '24/09/2020', 40, 82, 4.1, 126.1),
postgres-#     (10005, '24/07/2020', 40, 70, 3.5, 113.5),
postgres-#     (10008, '24/09/2020', 120, 187.5, 18.75, 326.25)
postgres-# ;
INSERT 0 3
postgres=#
```

1.3 Questions

1.3.1 Write a function for Energy charge calculation.

1.3.1.1 Command

```
CREATE or REPLACE FUNCTION calculate_energy_charge() RETURNS VOID as
$$
DECLARE
    c_cust_list CURSOR FOR
        SELECT cust_id, previous_reading, current_reading, customer_type
        FROM customer;
    c_cust_id customer.cust_id%type;
    c_previous_reading customer.previous_reading%type;
    c_current_reading customer.current_reading%type;
    c_customer_type customer.customer_type%type;
    units INT;
    energy_charge DECIMAL(10, 2);
BEGIN
    OPEN c_cust_list;
    LOOP
    FETCH c_cust_list INTO c_cust_id, c_previous_reading, c_current_reading,
        c_customer_type;
        EXIT WHEN NOT FOUND;
            units = c_current_reading - c_previous_reading;
            IF units > 200 THEN
                IF c_customer_type = 'A' THEN
                    energy_charge = 1.75 * units;
                ELSIF c_customer_type = 'I' THEN
                    energy_charge = 1.5 * units;
                ELSIF c_customer_type = 'C' THEN
                    energy_charge = 2 * units;
                ELSIF c_customer_type = 'R' THEN
                    energy_charge = 1.6 * units;
                END IF;
            ELSIF units > 100 THEN
```

```
            IF c_customer_type = 'A' THEN
                energy_charge = 1 * 100 + (units - 100) * 1.25;
            ELSIF c_customer_type = 'I' THEN
                energy_charge = 1.25 * 100 + (units - 100) * 1.3;
            ELSIF c_customer_type = 'C' THEN
                energy_charge = 1.5 * 100 + (units - 100) * 1.6;
            ELSIF c_customer_type = 'R' THEN
                energy_charge = 1.3 * 100 + (units - 100) * 1.4;
            END IF;
        ELSE
            IF c_customer_type = 'A' THEN
                energy_charge = 1 * units;
            ELSIF c_customer_type = 'I' THEN
                energy_charge = 1.25 * units;
            ELSIF c_customer_type = 'C' THEN
                energy_charge = 1.5 * units;
            ELSIF c_customer_type = 'R' THEN
                energy_charge = 1.3 * units;
            END IF;
        END IF;

        INSERT INTO bill(cust_id, energy_charge)
        VALUES (c_cust_id, energy_charge);
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT FROM calculate_energy_charge();
```

1.3.1.2 Output

```
postgres=# SELECT FROM calculate_energy_charge();
--
(1 row)

postgres=# SELECT * FROM bill;
 cust_id | bill_date  | fixed_charge | energy_charge | sur_charge | net_amount
---------+------------+--------------+---------------+------------+------------
   10005 | 2020-09-24 |        40.00 |         82.00 |       4.10 |     126.10
   10005 | 2020-07-24 |        40.00 |         70.00 |       3.50 |     113.50
   10008 | 2020-09-24 |       120.00 |        187.50 |      18.75 |     326.25
   10005 |            |              |         65.00 |            |
   10004 |            |              |        292.40 |            |
   10003 |            |              |        125.00 |            |
   10008 |            |              |        525.00 |            |
   10002 |            |              |        233.20 |            |
(8 rows)

postgres=#
```

1.3.2 Write a function for Net amount calculation in Bill table. Use a cursor to update the 'Net amount' and 'Bill date' (as today's date) in the Bill table.

1.3.2.1 Command

```
CREATE or REPLACE FUNCTION calculate_net_amount() RETURNS VOID as
$$
DECLARE
    c_cust_list CURSOR FOR
        SELECT c.cust_id, energy_charge, bill_date, meter_type FROM
            customer AS c, bill AS b WHERE c.cust_id = b.cust_id;
    c_cust_id customer.cust_id%type;
    c_energy_charge bill.energy_charge%type;
    c_bill_date bill.bill_date%type;
    c_meter_type customer.meter_type%type;
    c_fixed_charge DECIMAL(10, 2);
    c_sur_charge DECIMAL(10, 2);
    net_charge DECIMAL(10, 2);
BEGIN
    OPEN c_cust_list;
    LOOP
    FETCH c_cust_list INTO c_cust_id, c_energy_charge, c_bill_date, c_meter_type;
```

```
        EXIT WHEN NOT FOUND;
            IF c_bill_date IS NULL THEN
                IF c_meter_type = 1 THEN
                    c_fixed_charge = 40 + 17;
                    c_sur_charge = 0.05 * c_energy_charge;
                ELSE
                    c_fixed_charge = 120 + 17;
                    c_sur_charge = 0.1 * c_energy_charge;
                END IF;
                net_charge = c_energy_charge + c_fixed_charge + c_sur_charge;

                UPDATE bill
                SET net_amount = net_charge, bill_date = CURRENT_DATE,
                    fixed_charge = c_fixed_charge, sur_charge = c_sur_charge
                WHERE cust_id = c_cust_id AND bill_date IS NULL;
            END IF;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT FROM calculate_net_amount();

SELECT * FROM bill;
```

1.3.2.2 Output

```
postgres=# SELECT FROM calculate_net_amount();
--
(1 row)

postgres=#
postgres=# SELECT * FROM bill;
 cust_id | bill_date  | fixed_charge | energy_charge | sur_charge | net_amount
---------+------------+--------------+---------------+------------+------------
   10005 | 2020-09-24 |        40.00 |         82.00 |       4.10 |     126.10
   10005 | 2020-07-24 |        40.00 |         70.00 |       3.50 |     113.50
   10008 | 2020-09-24 |       120.00 |        187.50 |      18.75 |     326.25
   10005 | 2020-11-24 |        57.00 |         65.00 |       3.25 |     125.25
   10004 | 2020-11-24 |       137.00 |        292.40 |      29.24 |     458.64
   10003 | 2020-11-24 |        57.00 |        125.00 |       6.25 |     188.25
   10008 | 2020-11-24 |       137.00 |        525.00 |      52.50 |     714.50
   10002 | 2020-11-24 |       137.00 |        233.20 |      23.32 |     393.52
(8 rows)

postgres=#
```

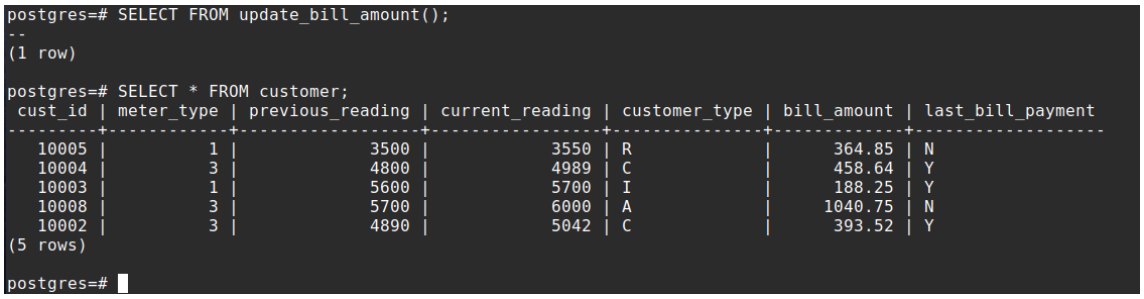
1.3.3 Write a procedure to update the 'Bill amount' in the Customer table.

1.3.3.1 Command

```
CREATE OR REPLACE PROCEDURE updateBillAmount(
  id int
)
AS
$$
DECLARE
  amount float;
BEGIN
  SELECT COALESCE(SUM(net_amount),0) INTO amount FROM bill WHERE cust_id=id;

  UPDATE customer
  SET bill_amount = amount
  WHERE cust_id = id;

END;
$$
LANGUAGE PLPGSQL;
```

```
CREATE or REPLACE FUNCTION update_bill_amount() RETURNS VOID
AS $$
DECLARE
    c_cust_list CURSOR FOR
        SELECT * FROM customer;
    c_cust_id customer.cust_id%type;
BEGIN
    OPEN c_cust_list;
    LOOP
    FETCH c_cust_list INTO c_cust_id;
        EXIT WHEN NOT FOUND;
            CALL updateBillAmount(c_cust_id);
    END LOOP;
END;
$$
LANGUAGE plpgsql;

SELECT FROM update_bill_amount();
```

1.3.3.2 Output



```
postgres=# SELECT FROM update_bill_amount();
--
(1 row)

postgres=# SELECT * FROM customer;
 cust_id | meter_type | previous_reading | current_reading | customer_type | bill_amount | last_bill_payment
---------+------------+------------------+-----------------+---------------+-------------+-------------------
   10005 |          1 |             3500 |            3550 | R             |      364.85 | N
   10004 |          3 |             4800 |            4989 | C             |      458.64 | Y
   10003 |          1 |             5600 |            5700 | I             |      188.25 | Y
   10008 |          3 |             5700 |            6000 | A             |     1040.75 | N
   10002 |          3 |             4890 |            5042 | C             |      393.52 | Y
(5 rows)

postgres=#
```

1.3.4 Create a trigger (to display a message “Previous dues for customer is <amount>”) whenever the Bill amount is updated for a customer whose Last Bill payment = ‘N’.

1.3.4.1 Command

```
CREATE OR REPLACE FUNCTION show_dues() RETURNS TRIGGER
AS
$$
BEGIN
IF (OLD.last_bill_payment = 'N') THEN
    raise info 'Previous dues for customer is %', OLD.bill_amount;
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;


CREATE TRIGGER dues_trigger
AFTER UPDATE OF bill_amount ON customer
FOR EACH ROW
EXECUTE PROCEDURE show_dues();
```

1.3.4.2 Output

```
postgres=# CREATE OR REPLACE FUNCTION show_dues() RETURNS TRIGGER
postgres-# AS
postgres-# $$
postgres$# BEGIN
postgres$#  IF (OLD.last_bill_payment = 'N') THEN
postgres$#     raise info 'Previous dues for customer is %', OLD.bill_amount;
postgres$#     END IF;
postgres$#
postgres$#     RETURN NEW;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=#
postgres=#
postgres=# CREATE TRIGGER dues_trigger
postgres-# AFTER UPDATE OF bill_amount ON customer
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE show_dues();
CREATE TRIGGER
postgres=#
postgres=# SELECT FROM update_bill_amount();
INFO:  Previous dues for customer is 364.85
INFO:  Previous dues for customer is 1040.75
--
(1 row)
```

1.4 Final Tables

1.4.1 Customer Table

```
postgres=# SELECT FROM update_bill_amount();
--
(1 row)

postgres=# SELECT * FROM customer;
 cust_id | meter_type | previous_reading | current_reading | customer_type | bill_amount | last_bill_payment
---------+------------+------------------+-----------------+---------------+-------------+-------------------
   10005 |          1 |             3500 |            3550 | R             |      364.85 | N
   10004 |          3 |             4800 |            4989 | C             |      458.64 | Y
   10003 |          1 |             5600 |            5700 | I             |      188.25 | Y
   10008 |          3 |             5700 |            6000 | A             |     1040.75 | N
   10002 |          3 |             4890 |            5042 | C             |      393.52 | Y
(5 rows)

postgres=#
```

1.4.2 Bills table

```
postgres=# SELECT FROM calculate_net_amount();
--
(1 row)

postgres=#
postgres=# SELECT * FROM bill;
 cust_id | bill_date  | fixed_charge | energy_charge | sur_charge | net_amount
---------+------------+--------------+---------------+------------+------------
   10005 | 2020-09-24 |        40.00 |         82.00 |       4.10 |     126.10
   10005 | 2020-07-24 |        40.00 |         70.00 |       3.50 |     113.50
   10008 | 2020-09-24 |       120.00 |        187.50 |      18.75 |     326.25
   10005 | 2020-11-24 |        57.00 |         65.00 |       3.25 |     125.25
   10004 | 2020-11-24 |       137.00 |        292.40 |      29.24 |     458.64
   10003 | 2020-11-24 |        57.00 |        125.00 |       6.25 |     188.25
   10008 | 2020-11-24 |       137.00 |        525.00 |      52.50 |     714.50
   10002 | 2020-11-24 |       137.00 |        233.20 |      23.32 |     393.52
(8 rows)

postgres=#
```

1.5 Result

Programs were implemented using Postgresql and outputs were verified.